Código que facilita los cálculos de una nivelación topográfica

Code to facilitate topographic leveling calculations

Tejeda Campos Dayra Nataly (1) Vargas Méndez Ilse Lilith (2)

(1) Facultad de Ingeniería Civil, Campus Coquimatlán, Colima - Coquimatlán Kilómetro 9, Jardines del Llano, 28400 Coquimatlán, Col. dtejeda1@ucol.mx, (2) ivargas@ucol.mx.

Resumen

En el presente trabajo se producirá un código de Python que tendrá como finalidad leer datos obtenidos previamente en campo de una nivelación topográfica, para la facilitación de sus cálculos para así lograr obtener, la altura de instrumento, cotas y principalmente el desnivel.

Palabras clave: Código, nivelación, cálculos, desnivel.

Abstract

In the present work a Python code will be produced with the purpose of reading data previously obtained in field of a topographic leveling, for the facilitation of its calculations to obtain the instrument height, elevations and mainly the slope.

Keywords: Code, leveling, calculations, slope.

• Introducción

La nivelación topográfica, es un método altimétrico, que tiene como objetivo obtener la cota de uno o varios puntos, a través de observaciones topográficas como ángulos, desniveles, distancias y un conjunto de mediciones y cálculos para asignar a un punto de cota, con relación a un plano de referencia determinado. En este trabajo se elaborará un código que facilite los cálculos de una nivelación para obtener la altura del instrumento, cotas y el desnivel. Para ello, se hará una breve investigación de los cálculos de cada punto a obtener en el código de Python.

- **Desarrollo**

Python es un lenguaje de programación de alto nivel, interpretado y de alto propósito; es uno de los lenguajes más usados para el desarrollo de software.

La elección de Python se debe a que se trata de un lenguaje de programación fácil de aprender y potente. Tiene eficaces estructuras de datos de alto nivel y una solución de programación orientada a objetos eficaz. La elegante sintaxis de Python, su gestión de tipos dinámica y su naturaleza interpretada hacen de él el lenguaje ideal para guiones (scripts) y desarrollo rápido de aplicaciones, en muchas áreas y en la mayoría de las plataformas.

Python ofrece una amplísima colección de módulos (bibliotecas). Hay módulos para cualquier actividad imaginable: escritura de CGI, gestión de correo electrónico, desarrollo de interfaces gráficas de usuario, análisis de documentos HTML o XML, acceso a bases de datos, trabajo con expresiones regulares, etc.

Materiales

Los materiales utilizados para la realización de este proyecto son:

- Computadora
- Google Colaboratory
- Excel (datos de nivelación)

Pasos

1. Registramos los datos de una nivelación en un archivo csv.

P.V.	LEC (+)	A.I.	LEC (-)	COTA
BN-1	1.874			
PL1	2.108		0.912	
PL2	0.943		0.714	
BN-2			1.819	

Figura #4. Archivo csv de una nivelación.

2. En Google Colab abrimos un cuaderno nuevo para empezar a instalar las librerías que no teníamos necesitadas para el código.

Figura #2. Instalación de librerías.

3. Importamos las librerías requeridas.

```
import pandas as pd #modulo Pandas para lectura de diferente tipos de archivos
import numpy as np #modulo numpy, para manejo de arreglos y matrices
import folium as fl
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
```

Figura #3. Importación de librerías.

4. Del archivo csv que contiene los datos de una nivelación leemos los datos y los convertimos en un arreglo.

```
#Extracción de datos de un archivo excel
datos = pd.read_excel ('topogigio.xlsx', usecols=(1,2,3,4), skiprows=(0),dtype=float, sheet_name=0)
#Se hace lectura de un archivo xlsx, pero puede ser otro tipo de archivos como csv o txt, puedes leer la documentación adecuada
#sobre como extraer este tipo de archivos, así mismo existen foros donde ya vienen bloques programados para leer archivos
#el primer atributo es el nombre, se inserta como sting y en caso de que se encuentre en otro carpeta se debe poner la ruta a la
#carpeta
#el segundo atributo es sobre las columnas del archivo que se usarán
#el tercer atributo es para saltar filas, en este caso se omite la primera fila que es de encabezados
#el cuarto atributo es el inidce de la hoja a usar, 0 para la primera
datos = datos.values #convierte del formato de pandas a un arreglo

ndp = len(datos) #se obtiene el número de lecturas

datos[0,1] = datos[0,3]+datos[0,0] #el formato "nan" es problemático, por lo que se hace que este valor sea 0 para la altura de
# aparato en primera fila
```

Figura #4. Lectura del archivo csv y arreglo de datos.

5. Obtenemos la cota del puto de liga, la altura de aparato y calculamos el desnivel.

```
for i in range (1,ndp): #se omite la primera fila de la tabla y se sigue hasta la última
    datos[i,3] = datos[i-1,1] - datos[i,2] # se saca la cota del punto de liga
    datos[i,1] = datos[i,3]+datos[i,0] # se obtiene la altura de aparato

desnivel = np.sum(datos[:,0])-np.sum(datos[:,2]) # se obtiene el desnivel

import csv
with open('outfile.txt', 'w') as f:
    csv.writer(f, delimiter=' ').writerows(datos)

print("Desnivel = %.3f m" %(abs(desnivel)))
print(datos)
```

Figura #5. Calculo de la cota, altura del aparato y desnivel

```
Desnivel = 13.406 m
[[2.53000e-01 2.95603e+02 0.00000e+00 2.95350e+02]
 [3.55000e-01 2.92173e+02 3.78500e+00 2.91818e+02]
 [4.75000e-01 2.88723e+02 3.92500e+00 2.88248e+02]
 [4.96000e-01 2.85579e+02 3.64000e+00 2.85083e+02]
 [0.00000e+00 2.81944e+02 3.63500e+00 2.81944e+02]]
```

Figura #6. Resultado del código para la obtención del desnivel, cota y altura de instrumento.

6. Importamos librerías requeridas para un nuevo proceso.

```
import pandas as pd
import numpy as np
import folium as fl
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
```

Figura #7 Importación de librerías.

7. Procesamos coordenadas para poderlas representar en un mapa.

```
mapa=fl.Map()
mapa

mGye=fl.Map(location=[19.210330,-103.802482],width="60%",height="6
fl.Marker(location=[19.210330,-103.802482]).add_to(mGye)
fl.Marker(location=[19.210330,-103.802482],popup="Este punto marca
mGye
```

Figura #8. Procesamiento de coordenadas para la obtención de un mapa

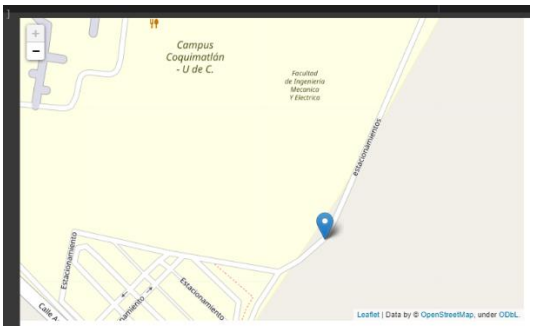

Figura #9. Resultado del mapa con las coordenadas de la nivelación.

- **<u>Manejo de datos</u>**

Este concepto hace referencia al conjunto de implementos funcionales que te ayudarán a codificar todo este lenguaje de programación para crear una interfaz independiente.

Las librerías de Python son amplias y cuentan con gran cantidad de producciones en contenidos. Constan de diversos módulos que permiten el acceso de funcionalidades específicas del sistema como entrada y salida de archivos, soluciones estandarizadas a problemas de programación, etc.

Además, dependiendo del sistema operativo que tengas, puedes conseguir diferentes funciones de cada una de las librerías de Python. Por ejemplo, para el sistema Windows se incluye la biblioteca estándar completa junto con componentes adicionales.
Un plus para las librerías de Python es que cuentan con una colección de componentes como, por ejemplo, programas individuales, módulos, paquetes, frameworks, aplicaciones y más funciones que puedes encontrar en Python Package Index..

Adentrandonos a las librerías que utilizaremos son las siguientes:

- *<u>Map</u>*

<u>C++ map estándar</u>

Map es un contenedor asociativo para contener en orden una lista de parejas de valores únicos asociados como clave/valor. En orden de poder crear objetos maps en nuestros programas deberemos incluir el uso de la clase map mediante la expresión: Otro aspecto que se debe de entender acerca de la estructura map es que esta está organizada para contener elementos asociados en parejas, de ahí, la necesidad de entender el comportamiento de la plantilla pair.

<u>C++ pair estándar</u>

pair es una plantilla cuyo propósito es contener una pareja de valores.

los miembros de pair son dos, first es el nombre del primero de ellos y el segundo se llama second. Cada uno de los miembros de pair pueden ser de tipos diferentes. Para poder

usar la plantilla pair en nuestro programa se debe de incluir la directiva: pair es una estructura independiente y puede ser usada con diversos fines, sin embargo, la importancia de pair radica en el hecho de que esta es usada como estructura elemental para construir contenedores tipo MAPS.

Antes de mostrar ejemplos de map mostraremos un par de ejemplos del uso de la estructura pair. Así, el siguiente programa utiliza pair para crear una pareja de valores asociados como (cadena, numero).

- *Basemap*

La librería Basemap para producir mapas con Python

Con este conjunto de herramientas podremos representar información geográfica vectorial y ráster sobre gráficos 2D. Funciona sobre diversas librerías además de Matplotlib como Proj4, Generic Mapping Tools y GEOS.

- *Matplotlib*

Es una de las principales librerías para visualización y análisis de datos de cualquier tipo, también espaciales.

Matplotlib es una librería de Python especializada en la creación de gráficos en dos dimensiones.

Permite crear y personalizar los tipos de gráficos más comunes, entre ellos:

- Diagramas de barras
- Histograma
- Diagramas de sectores
- Diagramas de caja y bigotes
- Diagramas de violín
- Diagramas de dispersión o puntos
- Diagramas de líneas
- Diagramas de áreas
- Diagramas de contorno
- Mapas de color

y combinaciones de todos ellos.

En la siguiente galería de gráficos pueden apreciarse todos los tipos de gráficos que pueden crearse con esta librería.

Creación de gráficos con matplotlib

Para crear un gráfico con matplotlib es habitual seguir los siguientes pasos:

1. Importar el módulo pyplot.
2. Definir la figura que contendrá el gráfico, que es la region (ventana o página) donde se dibujará y los ejes sobre los que se dibujarán los datos. Para ello se utiliza la función subplots().
3. Dibujar los datos sobre los ejes. Para ello se utilizan distintas funciones dependiendo del tipo de gráfico que se quiera.
4. Personalizar el gráfico. Para ello existen multitud de funciones que

permiten añadir un título, una leyenda, una rejilla, cambiar colores o personalizar los ejes.
5. Guardar el gráfico. Para ello se utiliza la función savefig().
6. Mostrar el gráfico. Para ello se utiliza la función show().

- *Pandas*

Pandas es una muy popular librería de código abierto dentro de los desarrolladores de Python, y sobre todo dentro del ámbito de Data Science y Machine Learning, ya que ofrece unas estructuras muy poderosas y flexibles que facilitan la manipulación y tratamiento de datos.

Pandas surgió como necesidad de aunar en una única librería todo lo necesario para que un analista de datos pudiese tener en una misma herramienta todas las funcionalidades que necesitaba en su día a día, como son: cargar datos, modelar, analizar, manipular y prepararlos.

Las principales características de esta librería son:

- Define nuevas estructuras de datos basadas en los arrays de la librería NumPy pero con nuevas funcionalidades.
- Permite leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL.
- Permite acceder a los datos mediante índices o nombres para filas y columnas.
- Ofrece métodos para reordenar, dividir y combinar conjuntos de datos.
- Permite trabajar con series temporales.
- Realiza todas estas operaciones de manera muy eficiente.

Tipos de datos de Pandas

Pandas dispone de tres estructuras de datos diferentes:

- Series: Estructura de una dimensión.
- DataFrame: Estructura de dos dimensiones (tablas).
- Panel: Estructura de tres dimensiones (cubos).

```
import pandas as pd
s = pd.Series(['Matemáticas', 'Historia', 'Economía', 'Programación', 'Inglés'], dtype='string')
print(s)
Matemáticas
Historia
Economía
Programación
Inglés
```

Estas estructuras se construyen a partir de arrays de la librería NumPy, añadiendo nuevas funcionalidades.

La clase de objetos Series

Son estructuras similares a los arrays de una dimensión. Son homogéneas, es decir, sus elementos tienen que ser del mismo tipo, y su tamaño es inmutable, es decir, no se puede cambiar, aunque si su contenido.

Dispone de un índice que asocia un nombre a cada elemento de la serie, a través de la cual se accede al elemento.

Ejemplo. La siguiente serie contiene las asignaturas de un curso.

Índice →	A1	A2	A3	A4
Valores →	Matemáticas	Economía	Programación	Inglés

Creación de series

Creación de una serie a partir de una lista

- Series (data=lista, index=indices, dtype=tipo): Devuelve un objeto de tipo Series con los datos de la lista $lista$, las filas especificadas en la lista índices y el tipo de datos indicado en tipo. Si no se pasa la lista de índices se utilizan como índices los enteros del 0 al $n-1$, done n es el tamaño de la serie. Si no se pasa el tipo de dato se infiere.

Creación de una serie a partir de un diccionario

- Series (data=diccionario, index=indices): Devuelve un objeto de tipo Series con los valores del diccionario $diccionario$ y las filas especificadas en la lista $indices$. Si no se pasa la lista de índices se utilizan como índices las claves del diccionario.

```
import pandas as pd
s = pd.Series({'Matemáticas': 6.0,  'Economía': 4.5, 'Programación': 8.5})
print(s)
Matemáticas    6.0
Economía       4.5
Programación   8.5
```

Folium es una librería de Python que se usa para visualizar mapas, Folium nace de la unión de otro lenguaje muy utilizado en SIG llamado JavaScript que es usado en entornos web y su librería Leaflet y el lenguaje Python. Mediante Folium podemos manipular dichos mapas de Leaflet con Python.

Respecto a su funcionamiento, Folium crea el vínculo entre Datasets que contienen datos cartográficos de diferentes objetos que son manipulados con Python y Leaflet que permiten generar la visual de la cartografía. También es posible, por ejemplo, ubicar en un mapa las diferentes ubicaciones de un restaurante en una ciudad usando un dataset que contenga sus coordenadas de GPS. Los mapas producidos de esta forma contienen una capa de base y otras capas generadas por Folium que se superponen.

La cantidad y variedad de objetos que se pueden visualizar con Folium es muy grande. Entre ellos podemos mencionar, mapas de diferentes tipos, igual de numerosos que los objetos de tipo vectoriales (círculo, polígono, rectangular, pin, etc), o incluso grillas que permiten crear ciertos bordes, por ejemplo.

Folium es una de las librerías más asequibles para análisis y representación geoespacial, por la sencillez de la sintaxis.

Una de las características que más atrae es que no necesita convertir el DataFrame en un GeoDataFrame. Solo teniendo una serie con la latitud y otra serie con la longitud del punto

espacial, es suficiente para que Folium lo represente en un mapa.

En cualquiera de las representaciones con Folium el primer paso es desarrollar el mapa sobre el que se va a pintar el resto de la información. Para ello, hay que centrarlo en un punto con latitud y longitud.

Folium utiliza por defecto como tipo de mapa base OpenStreetMap, pero hay otras opciones (es el parámetro que corresponde a “tiles”).

Normalmente, se selecciona el primer registro del dataset como referencia del centro del mapa.

Usando folium.Map() , crearemos un mapa base y lo almacenaremos en un objeto. Esta función toma coordenadas de ubicación y valores de zoom como argumentos.

Sintaxis: folium.Map (ubicación, mosaicos = “OpenStreetMap” zoom_start = 4)

Parámetros:

- Ubicación: lista de coordenadas de ubicación
- Mosaicos: el valor predeterminado es OpenStreetMap. Otras Opciones: Tamen Terrain, Stamen Toner, Mapbox Bright, etc.
- Zoom_start: int

Código:

```
# import the folium, pandas libraries
import folium
import pandas as pd

# initialize the map and store it in a m object
m = folium.Map(location = [40, -95],
               zoom_start = 4)

# show the map
m.save('my_map.html')
```

Figura #10. Código que utiliza la librería Folium.

Producción:

Figura #11. Resultado del código que utiliza la Liberia Folium.

- **Resultados**

Después de obtener nuestra serie de datos en campo después de una nivelación, nos dimos a la tarea de diseñar un código que nos facilitara los cálculos que hacemos normalmente para obtener tanto la altura del instrumento, cotas e incluso el desnivel.

Antes de empezar instalamos todas las librerías que vamos a utilizar en cada una de las partes.

Figura #12. Resultado de instalar las librerías correspondientes.

Lo primero que vamos a hacer es leer nuestros datos previamente registrados en Excel, nos damos cuenta de que nos hacen falta las dos columnas que son las que vamos a obtener que son la I.H y la cota, más aparte que obtendremos el desnivel.

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 topografy='topogigio.xlsx'
4 df= pd.read_excel(topografy)
5 print(df.head())
6
```

```
   P.O  LECT +  I.H.  LECT -    COTA
0  BN1   0.253   NaN   0.000  295.35
1  PL1   0.355   NaN   3.785    0.00
2  PL2   0.475   NaN   3.925    0.00
3  PL3   0.496   NaN   3.640    0.00
4  BN2   0.000   NaN   3.635    0.00
```

Figura #13. Obtención de la I.H. y Cota

(En anexo) Figura #14. Código para la obtención del desnivel

Nuestro código principal empieza dándonos una ruta, que es lo primero que tenemos que hacer es crearla para obtener la lectura y extracción de datos de nuestro archivo de Excel. Se hace esta lectura, pero nuestro código se diseño para que pueda leer otro tipo de archivo, tanto como csv o txt siempre y cuando tenga la información ordenada y adecuada. De donde vamos a extraer el archivo estaremos utilizando la librería os donde ya vienen bloques programados para leer este tipo de archivos.

Continuando con el código nuestro primer atributo es el nombre que se insertara como Sting, pero si en dado caso de que se encuentre en otro sitio se debe poner una nueva ruta a ello.

El segundo atributo es acerca de las columnas que se estarán utilizando.

El tercer atributo es para saltar filas, en nuestro caso se omitirá la primera fila que es de encabezados.

Nuestro cuarto atributo es el índice de la hoja a usar que en este caso es 0 para la primera,

Teniendo en cuenta que ya tenemos nuestra variable datos la vamos a convertir en un arreglo.

Luego obtenemos el número de lecturas, y prosigue a sacar los cálculos de las columnas restantes. Como se puede observar se obtuvo el desnivel y ya los resultados de nuestras partes faltantes.

```
27 desnivel = np.sum(datos[:,0])-np.sum(datos[:,2]) # se obtiene el desnivel
28
29 import csv
30 with open('outfile.txt', 'w') as f:
31     csv.writer(f, delimiter=' ').writerows(datos)
32
33 print("Desnivel = %.3f m" %(abs(desnivel)))
34 print(datos)
```

```
Desnivel = 13.406 m
[[2.53000e-01 2.95603e+02 0.00000e+00 2.95350e+02]
 [3.55000e-01 2.92173e+02 3.78500e+00 2.91818e+02]
 [4.75000e-01 2.88723e+02 3.92500e+00 2.88248e+02]
 [4.96000e-01 2.85579e+02 3.64000e+00 2.85083e+02]
 [0.00000e+00 2.81944e+02 3.63500e+00 2.81944e+02]]
```

Figura #15. Faltante del código y resultados de las columnas y el desnivel.

Integrando un poco la librería Folium decidimos agregar un mapa que nos muestre con detalle y especificamente donde se realizo la nivelacion, para ello debemos de insertar las cooredenas lo mas precisas posibles de nuestro primer punto. Que es 19.210330,-103.802482

```
import pandas as pd
import numpy as np
import folium as fl
import matplotlib.pyplot as plt

mapa=fl.Map()
mapa

mGye=fl.Map(location=[19.210330,-103.802482],width="60%",height="60%", zoom_start=25)
fl.Marker(location=[19.210330,-103.802482]).add_to(mGye)
fl.Marker(location=[19.210330,-103.802482],popup="Este punto marca donde se hizo la nivelación").add_to(mGye)
mGye
```

Figura #16. Código para la obtención del punto con nuestras coordenadas.

Teniendo como resultado nuestro punto exacto expresado en el mapa:

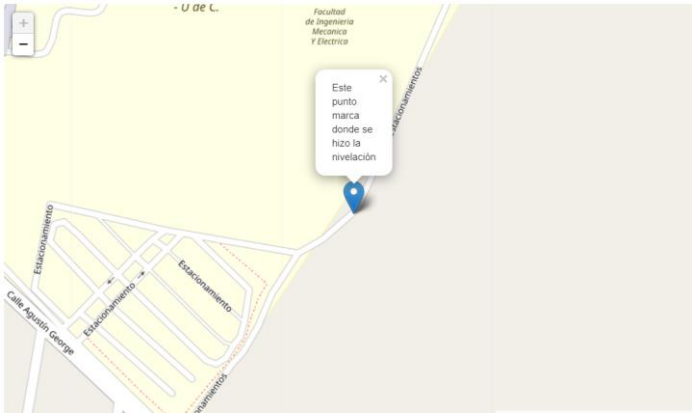

Figura #17. Mapa con el punto con nuestras coordenadas.

- **<u>Conclusiones</u>**

La programación trasciende todos los lenguajes, sobre todo porque también es una excelente herramienta para la resolución de problemas mediante la lógica y el ingenio. Es el arte de desarrollar soluciones a paradigmas complejos desde cero, basándose en el pensamiento estructural, lógico y creativo.

Con la realización de este proyecto podemos concluir que la programación puede ser de mucha utilidad a la hora de querer realizar cálculos para la obtención de información que sea de gran utilidad para nosotros a la hora de realizar algún trabajo topográfico en un futuro siendo Ingenieros Topógrafos Geomáticos.

Anexo

```python
import pandas as pd #modulo Pandas para lectura de diferente tipos de archivos
import numpy as np #modulo numpy, para manejo de arreglos y matrices
import folium as fl
import matplotlib.pyplot as plt


#Extracción de datos de un archivo excel
datos = pd.read_excel ('topogigio.xlsx', usecols=(1,2,3,4), skiprows=(0),dtype=float, sheet_name=0)
#Se hace lectura de un archivo xlsx, pero puede ser otro tipo de archivos como csv o txt, puedes leer la documentación adecuada
#sobre como extraer este tipo de archivos, así mismo existen foros donde ya vienen bloques programados para leer archivos
#el primer atributo es el nombre, se inserta como sting y en caso de que se encuentre en otro carpeta se debe poner la ruta a la
#carpeta
#el segundo atributo es sobre las columnas del archivo que se usarán
#el tercer atributo es para saltar filas, en este caso se omite la primera fila que es de encabezados
#el cuarto atributo es el ínidce de la hoja a usar, 0 para la primera
datos = datos.values #convierte del formato de pandas a un arreglo

ndp = len(datos) #se obtiene el número de lecturas

datos[0,1] = datos[0,3]+datos[0,0] #el formato "nan" es problemático, por lo que se hace que este valor sea 0 para la altura de aparato en
#primera fila

for i in range (1,ndp): #se omite la primera fila de la tabla y se sigue hasta la última
    datos[i,3] = datos[i-1,1] - datos[i,2] # se saca la cota del punto de liga
    datos[i,1] = datos[i,3]+datos[i,0] # se obtiene la altura de aparato

desnivel = np.sum(datos[:,0])-np.sum(datos[:,2]) # se obtiene el desnivel

import csv
with open('outfile.txt', 'w') as f:
    csv.writer(f, delimiter=' ').writerows(datos)

```

Figura #14. Código para la obtención del desnivel